Universidade Tecnológica Federal do Paraná
Curso de Engenharia Eletrônica

Eduardo Eugenio Rodrigues Sartor

Automação Residencial via IoT Utilizando a Plataforma NodeMCU

Toledo
2021

Eduardo Eugenio Rodrigues Sartor

Automação Residencial via IoT Utilizando a Plataforma NodeMCU

Home Automation via IoT Using the NodeMCU Platform

Trabalho de Conclusão de Curso apresentado à disciplina de Trabalho de Conclusão de Curso 2 do Curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná - UTFPR Campus Toledo, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador(a) Prof. Alessandro Paulo de Oliveira

Toledo
2021

Eduardo Eugenio Rodrigues Sartor

Automação Residencial via IoT Utilizando a Plataforma NodeMCU

Trabalho de Conclusão de Curso apresentado à disciplina de Trabalho de Conclusão de Curso 2 do Curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná - UTFPR Campus Toledo, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Toledo, 15 de Dezembro de 2021:

Prof. Alessandro Paulo de Oliveira
UTFPR-TD
Orientador

Prof. Felipe Walter Dafico Pfrimer
UTFPR-TD

Prof. Jose Dolores Vergara Dietrich
UTFPR-TD

Toledo
2021

A folha de aprovação assinada encontra-se na coordenação do curso.

RESUMO

A proposta deste trabalho é reunir comandos (acionamento de lâmpadas, motores e um buzzer) e dados (valores de temperatura e umidade ambiente) em um aplicativo *mobile*, onde é possível fazer o controle desses comandos, além de monitorar os dados obtidos. Estas atividades são normalmente realizadas manualmente e presencialmente e com a implementação deste projeto passam a ficar disponíveis ao usuário de qualquer lugar onde se tenha acesso a internet, gerando um maior conforto, economia e segurança para o usuário. Visando melhorar a visualização do funcionamento do sistema, o monitoramento dos dados e acionamento dos comandos foram realizados em uma maquete de representação residencial. Foi utilizado no sistema de automação a plataforma NodeMCU como microcontrolador, a qual estabelece a conexão entre sensores, atuadores, aplicativo *mobile* e o banco de dados hospedado no *framework Firebase* via conceito de Internet das Coisas.

Palavras-chave: Automação Residencial. Internet das Coisas. NodeMCU. Firebase. Flutter.

ABSTRACT

The purpose of this work is to gather commands (activation of lamps, motors and a buzzer) and data (temperature and ambient humidity values) in a *mobile* application, where it is possible to control these commands, in addition to monitoring the data obtained. These activities are normally carried out manually and in person, and with the implementation of this project, they become available to the user from any place where internet access is available, generating greater comfort, economy and security for the user. Aiming to improve the visualization of system operation, data monitoring and command activation were performed in a mockup of residential representation. The NodeMCU platform was used in the automation system as a microcontroller, which establishes the connection between sensors, actuators, *mobile* application and the database hosted in *Firebase framework* via the Internet of Things concept.

Keywords: Home automation. Internet of Things. NodeMCU. Firebase. Flutter

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

A	Ampère
A/D	Analógico/Digital
APK	*Android Application Pack*
C	Celsius
CC	Corrente Contínua
FPGA	*Field Programmable Gate Array*
GND	*Ground*
GPIO	*General Purpose Input/Output*
Hz	Hertz
I/O	*Input/Output*
IDE	*Integrated Development Environment*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
IoT	*Internet of Things*
IP	*Internet Protocol*
MDF	*Medium Density Fiberboard*
NA	Normalmente aberto
NC	*Normally Closed*
NF	Normalmente fechado
NFC	*Near Field Communication*
NO	*Normally Open*
NTC	*Negative Temperature Coefficient*
OWL	*Web Ontology Language*
PTC	*Positive Temperature Coefficient*
PWM	*Pulse Width Modulation*
RDF	*Resource Description Framework*
RFID	*Radio-Frequency Identification*
UART	*Universal Synchronous Receiver/Transmitter*
UI	*User Interface*
UR	Umidade Relativa

USB	*Universal Serial Bus*
V	Volt
Vin	Tensão de entrada
WPAN	*Wireless Personal Area Network*

SUMÁRIO

1 INTRODUÇÃO

Uma casa inteligente é um ambiente doméstico que possui automatização das suas tarefas cotidianas através de aparelhos eletroeletrônicos que são conectados a um sistema de controle. A automação de uma casa inclui o controle da iluminação dos cômodos, controle de temperatura, ventilação, dentre outros segmentos. Neste trabalho foram abordados o controle da iluminação, acionamento de motores (persianas e portão eletrônico) e monitoramento de sensores (temperatura, umidade, movimento e presença).

A automação residencial visa aumentar o conforto, a eficiência no consumo energético e também a segurança dos ambientes. Através de um sistema de monitoramento e controle unificado, as tarefas poderão ser acionadas pelos usuários de maneira remota (*smartphone*, computador, *smartwatches*), ou então pelo próprio sistema, caso programado para tomar uma decisão a partir dos dados coletados (SAIKRISHNA; VIJAYKIRAN, 2017).

Apesar do conforto e praticidade serem os principais motivos para aderir a automação em uma residência, a economia de recursos vem sendo mais comum no cotidiano das pessoas devido ao aumento considerável no preço da energia. Nesse contexto, ao possuir um sistema que controle e, consequentemente, melhore o consumo energético, será vantajoso não somente para os consumidores, mas também para toda a rede elétrica e o meio ambiente através do conceito de sustentabilidade (SAIKRISHNA; VIJAYKIRAN, 2017).

Pensando nisso, esse trabalho foi desenvolvido para aplicar os conceitos emergentes sobre Internet das Coisas (IoT), em um sistema de automação residencial com baixo custo financeiro e energético através de um aplicativo intuitivo e de fácil utilização.

2 OBJETIVOS

Este trabalho possui como objetivo principal, realizar o estudo e implementação de uma automação residencial através da programação da plataforma NodeMCU. Esta, por sua vez, realiza a comunicação com o aplicativo *mobile* e através do conceito de Internet das Coisas, habilita o controle e o monitoramento remoto de certas atividades cotidianas e, desta forma, facilitando-as ao usuário.

2.1 OBJETIVOS ESPECÍFICOS

Para que fosse possível realizar o objetivo geral, o trabalho foi divido nos seguintes tópicos de estudo:

1. Criar um banco de dados no *framework Firebase* para armazenar os dados de monitoramento e controle do sistema.
2. Realizar a programação da plataforma NodeMCU de modo a fazer a conexão entre o banco de dados presente no *framework Firebase* e os sensores e atuadores.
3. Desenvolver um aplicativo *mobile* que realiza a comunicação do usuário com o restante do sistema.
4. Automatização das lâmpadas.
5. Monitoramento da temperatura e umidade.
6. Controle de persianas remotamente.
7. Controle de um portão eletrônico.
8. Monitoramento de presença e acionamento de alarme.
9. Integrar todos os sistemas isolados em um sistema completo final para representação na maquete.

3 JUSTIFICATIVA

A automação domina o setor industrial há um bom tempo, e vem se tornando cada vez mais presente no âmbito residencial. Entretanto, por ser uma área ainda considerada nova e emergente, a desinformação acaba por gerar certo receio na aquisição de um sistema automatizado. Isso, por sua vez, diminui a demanda desse serviço causando um aumento considerável no preço (THAMARAIMANALAN et al., 2018).

Porém, com a popularização do conceito Internet das Coisas (IoT) aliada ao desenvolvimento de tecnologias (padrão 5G para redes móveis e padrão WiFi 6 para redes locais) que expandem a acessibilidade de conexão a internet, é proporcionado um crescimento constante na busca e criação de materiais sobre automação residencial. Dessa forma, juntamente com levantamenos estatíscos (ver subseção 4.4.2) que apontam uma elevada movimentação econômica sobre o assunto, fica evidente o promissório futuro que os sistemas de automação residencial podem atingir. Dessa forma, faz-se necessário o estudo sobre alternativas de sistemas para automação que visem um baixo custo e com uma fácil implementação, que por sua vez, irão contribuir para sua popularização. Além disso, a automação remota facilita a vida do usuário em sua residência.

4 REFERENCIAL TEÓRICO

Para que fosse possível o desenvolvimento do trabalho, diferentes conteúdos vistos ao longo do curso tiveram que ser aplicados, como por exemplo: programação em C, microcontroladores, sensores e atuadores. Além dos conteúdos inerentes ao curso, o estudo sobre Internet das Coisas se fez necessário. Sendo assim, este capítulo contemplará uma revisão bibliográfica dos conceitos em questão.

4.1 MICROCONTROLADORES

Microcontroladores são circuitos integrados que realizam processamento de dados através do controle de operações lógicas. São programados através de *software* fazendo utilização de linguagem de programação, tendo como exemplo as linguagens C e Assembly. São constituídos de entradas e saídas digitais e analógicas, memórias RAM (*Random Access Memory*), ROM (*Read-Only Memory*), híbridas como *Flash* e EEPROM (*Eletrically-Erasable-Programmable Read-Only Memory*), unidade lógica e aritmética, temporizadores, portas de comunicação, registradores e toda estrutura interna que conecta esses elementos. Microcontroladores possuem grande utilização em sistemas embarcados inteligentes, nesse caso para automação residencial, devido ao seu baixo custo e simplicidade para implementação (TRINDADE; ALMEIDA; PINTO, 2009).

A Figura 1 apresenta o microcontrolador PIC16F877A da empresa Microchip, sendo esse modelo muito utilizado para o aprendizado do funcionamento e implementação de microcontroladores.

Figura 1 – Microcontrolador PIC16F877A

Fonte: Microchip (2020)

4.2 COMPONENTES ENVOLVIDOS NO SENSORIAMENTO E ATUAÇÃO

Nesta seção será realizado uma revisão sobre os componentes utilizados na aquisição dos sinais e posterior atuação, sendo esses: sensores, conversores, transmissores, controladores e atuadores.

4.2.1 SENSORES

A definição de sensor pode ser expandida em alguns segmentos, que se diferenciam pela função específica exercida.

Primeiramente, tem-se o elemento primário, sendo o sensor propriamente dito, é sensível a alguma forma de energia do ambiente e por isso está em contato direto com a variável física de interesse (temperatura, umidade, pressão, distância) (BRITO, 2014).

Normalmente, o sinal de saída de um sensor deve ser manipulado antes de ser enviado para o sistema de controle. Como por exemplo em uma célula piezoelétrica que, ao sofrer pressão mecânica, gera uma tensão elétrica de valor baixo. Controlar e realizar operações com um sinal de pequeno módulo intensifica o erro agregado à medição, sendo assim, é utilizado um amplificador que eleva o sinal de tensão para um nível que seja efetivo para a sua utilização e controle (WENDLING, 2010).

A Figura 2 mostra um exemplo de elemento primário, o sensor PT100, que realiza a aquisição da váriavel temperatura do ambiente e converte em um sinal elétrico.

Figura 2 – Sensor de Temperatura PT100

Fonte: View Tech (2020)

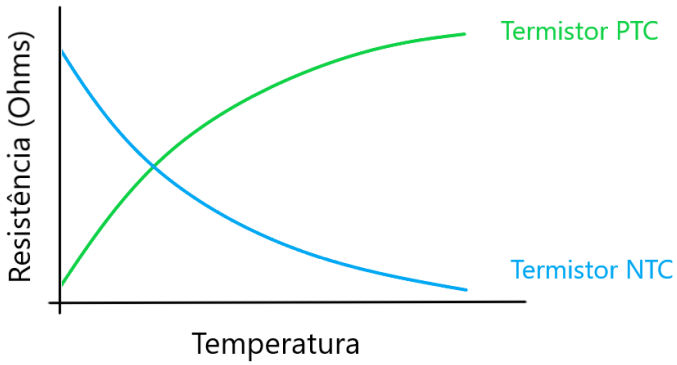
4.2.1.1 SENSORES ANALÓGICOS

São sensores em que, ao longo do tempo, sua grandeza elétrica pode assumir infinitos valores dentro de uma faixa limitada, desde que respeitando a sua faixa de operação. Exemplos de sensores analógicos são os sensores termistores NTC (*Negative Temperature*

Coefficient) e PTC (*Positive Temperature Coefficient*) , que são resistores cuja resistência varia conforme a temperatura do ambiente em que está inserido (WENDLING, 2010).

A Figura 3 mostra o gráfico que representa a variação da resistência dos termistores em função da temperatura.

Figura 3 – Gráfico da resposta dos termistores NTC e PTC

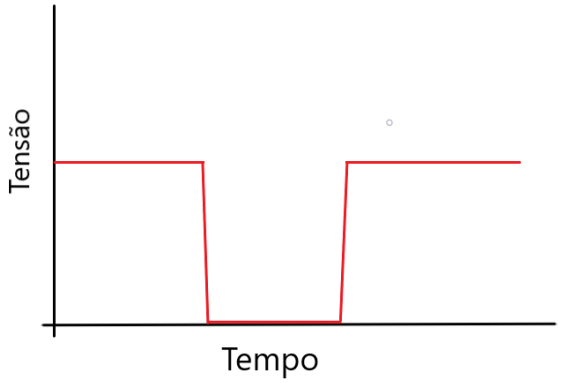


Fonte: Autoria própria

4.2.1.2 SENSORES DIGITAIS

Já os sensores digitais, possuem na sua saída um sinal que podem assumir somente valores binários, sendo comumente interpretados como Alto (em inglês *High*) ou Baixo (em inglês *Low*), ou simplesmente um e zero. Como não existe alguma grandeza física que possua esse comportamento, possuem um comparador que converte o sinal para o sistema de controle. São aplicados frequentemente no setor industrial como, por exemplo, na detecção da passagem de objetos por uma esteira, fazendo assim a contabilidade de itens ou então reconhecendo uma falha (PATSKO, 2006a).

Figura 4 – Gráfico da resposta de um sensor digital genérico



Fonte: Autoria própria

O gráfico visto na Figura 4 representa, de maneira genérica, a resposta em tensão de um sensor digital, no qual os valores são bem definidos para os intervalos existentes.

4.2.2 CONVERSORES

Conversor é um tipo de transdutor que atua apenas com sinais padronizados e determinados. Um exemplo é o dispositivo da Figura 5 que pode converter um sinal de entrada de corrente (4-20 mA) ou tensão (0-5 V) em uma saída pneumática diretamente proporcional (3-9 psi)(BRITO, 2014).

Figura 5 – Conversor Eletropneumático série 500X

Fonte: Digitrol (2018)

4.2.3 TRANSMISSORES

Transmissor é um dispositivo que recebe o sinal do transdutor e o prepara para ser enviado a uma certa distância, respeitando um padrão de transmissão (THOMAZINI; ALBUQUERQUE, 2020).

4.2.4 CONTROLADORES

São dispositivos que realizam a comparação da variável de controle com o valor de ajuste padrão (*setpoint)*, e assim, caso necessário, fornecem um sinal para a correção da variável. Um exemplo muito comum de controlador dentro do nicho da automação é o CLP (Controlador Lógico Programável). A Figura 6 mostra um modelo de CLP presente no mercado (BRITO, 2014).

Figura 6 – Módulo CLP SIMATIC S7-1200

Fonte: Siemens (2020)

4.2.5 ATUADORES

São equipamentos presentes geralmente na etapa final de automação, sendo os responsáveis por realizar a conversão do sinal elétrico proveniente do sistema de controle, em alguma ação que atuará na variável controlada. São exemplos de atuadores: relés, válvulas hidráulicas, válvulas pneumáticas, motores (WENDLING, 2010).

Na Figura 7 pode ser visto um modelo de atuador para válvula de gás.

Figura 7 – Atuadores de gás Siemens SKP55 e SKP75

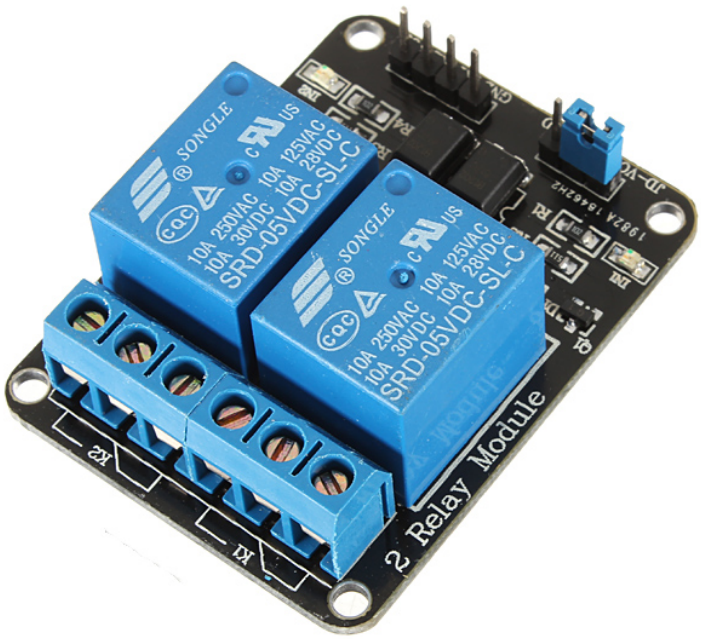
Fonte: Siemens (2020)

4.2.5.1 RELÉS

Pode-se definir um relé como um atuador eletromecânico, onde, através de um eletroimã, possui seus contatos comutados ao sofrer ação de uma determinada corrente elétrica. Os relés podem possuir duas configurações de funcionamento NF (normalmente fechado) e NA (normalmente aberto), ou em inglês NC (*Normally Closed*) e NO (*Normally Open*), respectivamente. Para o caso de um relé NF, seus contatos são mantidos fechados (em contato) por padrão, e ao ser energizado os contatos se abrem. Já um relé NA possui o funcionamento contrário, isto é, seus contatos são mantidos abertos e apenas quando energizado que o circuito é fechado (BRAGA, 2017).

A Figura 8 mostra um modelo comercial de módulo relé, onde dois relés são soldados em uma placa de circuito impresso possuindo seus circuitos de acionamento prontos para uso. Os módulos relés são comumente usados em projetos de microeletrônica, pois as placas no qual são soldados possuem as devidas proteções para trabalharem com corrente alternada de maneira segura.

Figura 8 – Módulo Relé 2 Canais



Fonte: FilipeFlop (2020)

4.3 MOTORES ELÉTRICOS

Motores elétricos são dispositivos que convertem energia elétrica em energia mecânica rotativa por meio de interações de fluxos magnéticos em seu interior. São constituídos primordialmente pelos seguintes elementos: rotor (parte rotativa) e estator (parte fixa). Os modelos mais difundidos de motores elétricos são os motores de corrente contínua (motores CC), os de corrente alternada (motores CA), os motores universais que funcionam para ambos os tipos de corrente, os servomotores e os motores de passo. Neste projeto foram utilizados dois tipos de motores, o motor CC e o motor de passo (PATSKO, 2006b).

4.3.1 MOTOR CC

Os motores CC possuem em sua composição física, além do estator e rotor, comutador e escovas.

Os motores CC são acionados por corrente contínua, caracterizando um sistema com campo magnético contínuo, porém com fluxo variável. O estator funciona como um ele-

troimã com suas polaridades fixas, dessa forma, quando as escovas entram em contato com o comutador a corrente passa pelos enrolamentos da bobina. Assim, devido à ação das forças magnéticas geradas entre a parte estática e a parte móvel, o rotor tende a girar na busca por uma nova condição de equilíbrio (MONTENEGRO, 2021).

4.3.2 MOTOR DE PASSO

Motores de passo são motores que convertem pulsos elétricos em movimentos mecânicos com variações angulares precisas. O eixo de um motor de passo é rotacionado em "passos", que são pequenos incrementos angulares, quando é aplicada uma determinada sequência de pulsos elétricos em seus terminais. A velocidade com que o eixo gira e o tamanho do ângulo rotacionado estão diretamente relacionados, respectivamente, com a frequência de pulsos recebidos e com o número de pulsos aplicados (SANTOS, 2008).

Algumas vantagens de se utilizar um motor de passo:

- Excelente resposta a aceleração e desaleração;
- Funcionamento excelente em baixa velocidade e elevado torque;
- Funcionamento nos dois sentidos de rotação.

4.4 INTERNET DAS COISAS

Internet das Coisas, expressão vinda do inglês *Internet of Things* (IoT), representa a intercomunicação entre objetos físicos e a internet. Essa conexão proporciona o acesso e controle remoto desses objetos e, através de sensores e atuadores, é moldada uma rede de objetos inteligentes que são capazes de realizar diversos processamentos, monitorar variavéis de controle e tomar decisões através das informações obtidas. A IoT é uma das principais tecnologias emergentes que abrange os mais diversos nichos comerciais, tais como o meio residencial, industrial, agrícola e urbano (PIRES et al., 2015).

O surgimento do conceito IoT é decorrente do avanço de variadas áreas tecnológicas tais como sistemas embarcados, microeletrônica, redes de sensores sem fio e comunicação móvel. De modo geral, esses avanços podem ser separados em três fases: a evolução do *hardware*; evolução do *software*; evolução da integração (ALBERTIN; ALBERTIN, 2017).

A evolução do *hardware* é referente aos equipamentos em si, isto é, o aperfeiçoamento da parte eletrônica, fazendo com que os componentes possuíssem maior capacidade de processamento e armazenamento ocupando um menor espaço físico, indo muito além dos computadores no seu sentido tradicional.

A evolução do *software* se deve pelo aumento das funcionalidades, facilidade de uso, diminuição do custo de processamento, padronização e popularização dos programas e aplicativos. Assim como o hardware, o uso de softwares passou a ser indispensável tanto

em organizações quanto pela população no geral, tornando-se uma infraestrutura natural do cotidiano humano

Já a evolução da integração, une os avanços de hardware e software e nos traz ao cenário atual, onde surge o conceito de IoT, pois agora, os aparelhos eletro/eletrônicos (*hardware*) conseguem se comunicar com uma rede através de programas de controle e gerenciamento (*softwares*)

4.4.1 BLOCOS DE CONSTRUÇÃO DA IOT

Para ser feito um plano de automação utilizando o conceito de IoT, alguns blocos são necessários na construção do sistema, sendo eles: identificação, comunicação, serviços, semântica, computação e sensores/atuadores (SANTOS et al., 2016).

4.4.1.1 IDENTIFICAÇÃO

Realiza a identificação dos objetos que serão conectados a rede, podendo ser utilizadas diferentes tecnologias de reconhecimento como RFID (*Radio-Frequency Identification*), NFC (*Near Field Communication*) e o mais comum e utilizado neste trabalho, endereçamento IP (*Internet Protocol*) .

4.4.1.2 COMUNICAÇÃO

Bloco referente às técnicas utilizadas para fazer a conexão entre os objetos da rede, sendo as mais comuns:

- **WiFi**: conexão sem fio de dispositivos, seguindo uma série de padrões de transmissão de codificação propostos pela norma IEEE 802.11
- **Bluetooth**: conexão do tipo WPAN (*Wireless Personal Area Network*)
- **IEEE 802.15.4**: padrão de conexão sem fio que efetua o controle de acesso para redes pessoais de baixa transmissão

4.4.1.3 SERVIÇOS

Uma rede de IoT pode oferecer diversos tipos de serviços, dentre esses, são destacados os Serviços de Identificação que realizam o mapeamento das Entidades Físicas (objetos, sensores, atuadores) de interesse em Entidades Virtuais (usuários conectados, sistema de controle); Serviços de Dados que fazem o armazenamento de dados coletados dos objetos da rede, como por exemplo amostras de temperatura durante certo período de análise; Serviços de Colaboração e Inteligência que atuam com uma tomada de decisão ao analisar certo cenário de dados, como por exemplo, fazer uma ligação para um celular quando a umidade de um silo estiver extrapolando os limites aceitáveis (SANTOS et al., 2016).

4.4.1.4 SEMÂNTICA

Trata-se da obtenção de conhecimento através de dados provenientes dos objetos presentes na rede. A partir disso, busca-se a melhora da eficiência dos recursos existentes. Para isso, são utilizadas técnicas como RDF (*Resource Description Framework*) , *Web Ontology Language (OWL)* e *Efficient XML Interchange (EXI)* (SANTOS et al., 2016).

4.4.1.5 COMPUTAÇÃO

Referente aos componentes eletrônicos da unidade de processamento como por exemplo: microcontroladores, processadores, FPGAs (*Field Programmable Gate Array*) . Esse bloco é responsável pelo controle dos objetos através da execução de algoritmos locais (JUNIOR; LIMA; OLIVEIRA, 2018).

4.4.1.6 SENSORIAMENTO

Bloco que realiza a coleta das informações do ambiente em que os objetos da rede estão inseridos, armazenam os dados e encaminham para o sistema microcontrolado, que ao fazer uma leitura das amostras, pode manipular o ambiente, através dos atuadores, conforme a tarefa que foram programados para exercer.

4.4.2 PERSPECTIVA GERAL FUTURA

Os rápidos avanços tecnológicos juntamente com a disseminação do conceito de IoT por parte de grandes empresas (Amazon, Google, etc.), gerou uma empolgação acerca do assunto. Segundo *McKinsey Global Insitute*, é estimado que a aplicação de IoT na economia global será de 4 % a 11 % do produto interno bruto mundial em 2025, cerca de 3,9 a 11,1 trilhões de dólares. Para o caso específico do Brasil a estimativa para 2025 é de 50 a 200 bilhões de dólares como impacto econômico anual (SEIXAS; CONTINI, 2017).

4.5 LINGUAGEM DE PROGRAMAÇÃO

Linguagem de programação é um método padronizado utilizado para expressar instruções (rotinas) de um programa a um computador, este que realizará a leitura do código compilado e será programado para realizar determinada tarefa. Pode-se citar como exemplo as seguintes linguagens: C, C++, Pascal, Python, Java, Lua (GOTARDO, 2015).

O microcontrolador ESP8266, utilizado neste projeto, é programado a partir de uma versão modificada da linguagem C, compilada e executada pelo Ambiente de Desenvolvimento Integrado do Arduino, conhecido do inglês como Arduino IDE (*Integrated Development Environment)*.

4.5.1 LINGUAGEM C

C é uma linguagem estruturada de médio nível, isso implica na permissão da manipulação de bits, bytes e endereços, elementos básicos para funcionamento dos computadores e microcontroladores. Um código realizado em C é usual pois possui alta portabilidade, isto é, sua adaptação de computador para computador é possível (SCHILDT, 1997).

4.5.2 LINGUAGEM DART

A linguagem *Dart* foi desenvolvida pela empresa Google com o objetivo inicial de substituir a linguagem padrão dos navegadores JavaScript. Porém, a linguagem *Dart* veio a se tornar uma excelente ferramenta para desenvolvimento rápido de aplicativos multiplataforma.

Algumas vantagens de se utilizar a linguagem *Dart* para a programação de aplicações *web* e *mobile*:

- Possui uma vasta gama de bibliotecas nativas que auxiliam na construção das aplicações;
- Integração com diversos recursos externos como por exemplo as plataformas *Firebase* e *Alexa*;
- O compilador possui tecnologia que permite o código ser executado em diferentes plataformas (nativa, *web*);
- Constantes atualizações e novas integrações sendo realizadas de maneira frequente;

5 MATERIAIS E MÉTODOS

Neste capítulo serão apresentados os materiais utilizados no desenvolvimento do trabalho e a programação realizada tanto no microcontrolador quanto no aplicativo *mobile*.

5.1 COMPONENTES E DISPOSITIVOS

Essa seção abordará os componentes utilizados na constituição do circuito, descrevendo suas especificações e funcionalidades dentro do projeto.

5.1.1 NODEMCU

O NodeMCU é uma plataforma aberta de desenvolvimento de IoT. Foi desenvolvido em 2014 por uma empresa chinesa chamada *Espressif Systems*, com a finalidade de prover um *firmware* (*software* que fornece controle de baixo nível para um *hardware* específico) para seu chip recém lançado na época, o ESP8266. Esse chip foi, em um primeiro momento, lançado como um microcontrolador com capacidade de acesso e comunicação utilizando WiFi. Porém, teve sua popularização com a criação do seu SDK (*Software Development Kit*) ou *devkit*, pois permitiu aos usuários que fizessem a programação direta ao chip sem que houvesse a necessidade de um microcontrolador. O *firmware* utiliza, originalmente, uma linguagem de programação chamada Lua, porém, neste trabalho será utilizado a IDE do Arduino, baseada na linguagem C, para realizar a compilação e gravação do código no chip através da biblioteca ESP8266 (VANAJA et al., 2018).

A Figura 9 mostra o modelo de SDK para o chip ESP8266 utilizado neste trabalho.

Figura 9 – NodeMCU ESP8266

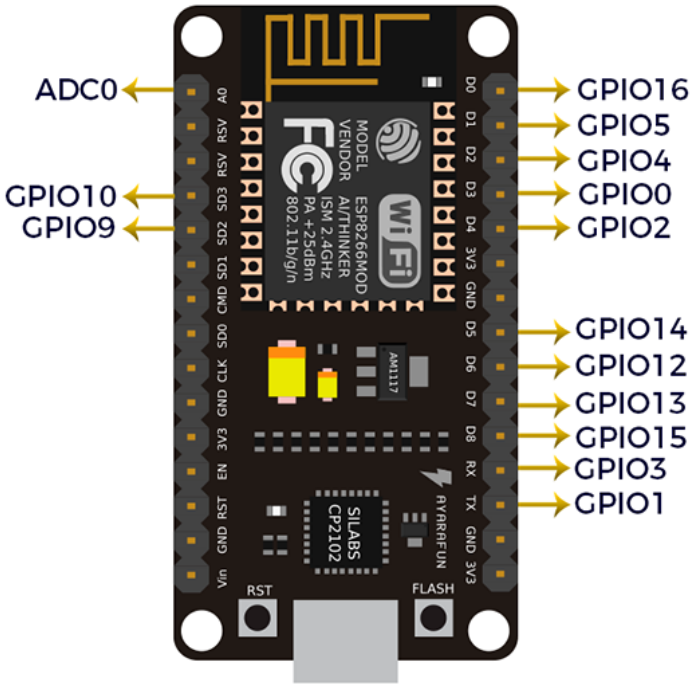
Fonte: Curto Circuito (2020)

Principais características da plataforma NodeMCU:

- Faixa de frequência: 2.4 GHz
- Taxa de transmissão: 110 à 460 Mbps
- Transmissão *wireless* nos padrões: IEEE 802.11b, IEEE 802.11g e IEEE 802.11n
- Alimentação por cabo Micro-USB: 4,0 a 9,0 V em tensão contínua
- Protocolo de comunicação: Serial UART
- Tensão lógica: 3,3 V
- Corrente consumida: mínimo de 70 mA (*standby*) e máximo de 220 mA
- Conversor A/D: 10 bits e Vin 0 1 V em tensão contínua
- GPIO (Portas lógicas utilizadas para controle e recepção de dados): 13 portas

A Figura 10 mostra o diagrama da correspondência das portas do NodeMCU para a IDE do arduíno.

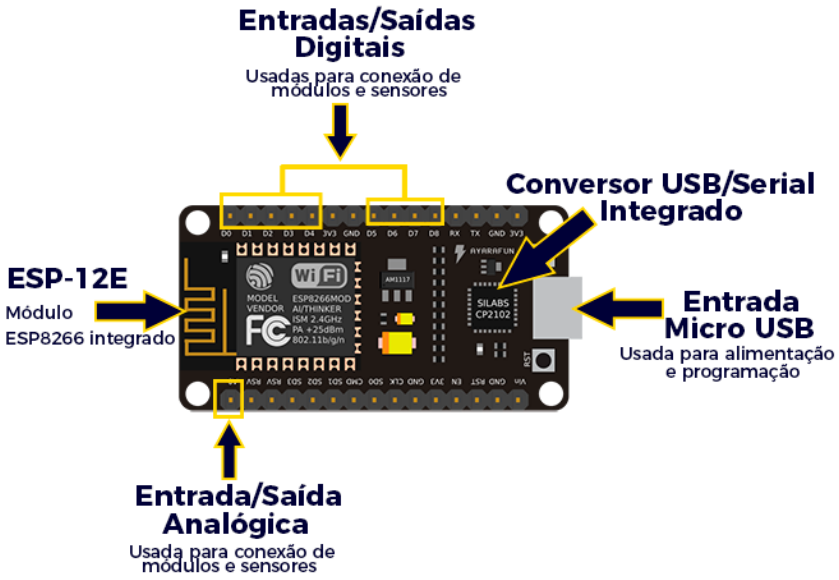
Figura 10 – Diagrama das portas digitais e analógica correspondentes NodeMCU - IDE Arduino



Fonte: Blog MasterWalker Shop (2016)

Na Figura 11 é possível visualizar a composição estrutural do kit de desenvolvimento NodeMCU.

Figura 11 – Diagrama da estrutura do NodeMCU

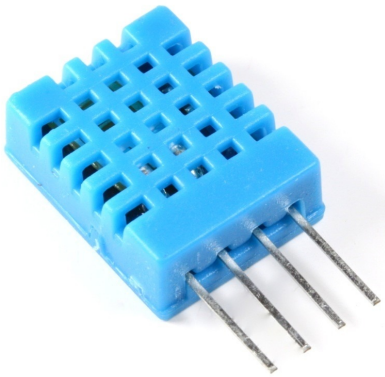


Fonte: Blog MasterWalker Shop (2016)

5.1.2 SENSOR DE UMIDADE E TEMPERATURA DHT11

A Figura 12 mostra o sensor de temperatura e umidade DHT11(*Digital Humidity and Temperature*). Possuindo um componente resistivo para mensurar umidade e um sensor do tipo NTC para a temperatura, o DHT11 gera em sua saída um sinal digital calibrado conforme a aquisição das variáveis à sua volta (SIPANI et al., 2018). Alguns pontos foram estudados para que fosse feita a escolha do sensor a ser utilizado nesse trabalho, e o DHT11 se mostrou viável a partir da seguinte análise: aceitável faixa de medição dos valores de temperatura e umidade para o projeto; alta confiabilidade e estabilidade nas medições; baixíssimo custo para aquisição; fácil implementação e suporte.

Figura 12 – Sensor DHT11



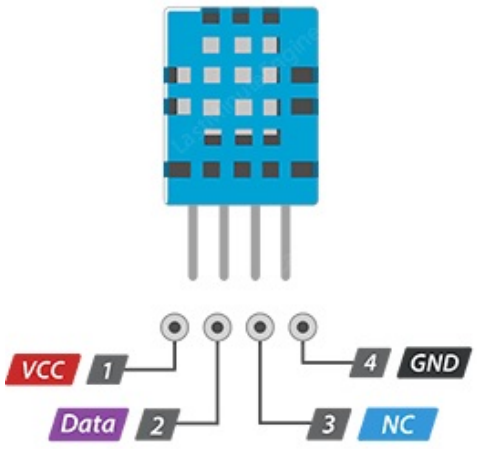
Fonte: FilipeFlop (2020)

Possui as seguintes características de operação:

- Faixa de medição de umidade: 20 a 90% UR
- Faixa de medição de temperatura: 0º a 50º C
- Alimentação: 3,3 a 5,5 V em tensão contínua
- Corrente consumida: 150 uA (*standby*) a 500 mA
- Precisão de medição da umidade: máximo 5,0% UR de diferença ao valor real
- Precisão de medição de temperatura: máximo 2,0º C de diferença ao valor real
- Tempo de resposta: 2 segundos

A relação dos pinos do sensor DHT11 é mostrada na Figura 13, sendo o pino 1 (VCC) a alimentação do sensor (3,3 a 5,5 V), o pino 2 (DATA) utilizado para fazer a comunicação com o microcontrolador através do protocolo *Serial interface (Single-Wire Two-Way)*, no caso, os valores mensurados de temperatura e umidade, o pino 3 (NC) não é utilizado, e por fim o pino 4 (GND) conectado ao *ground* (terra) do NodeMCU.

Figura 13 – DHT11 Pinout



Fonte: LastMinuteEngineers (2020)

5.1.3 SENSOR DE PRESENÇA PIR

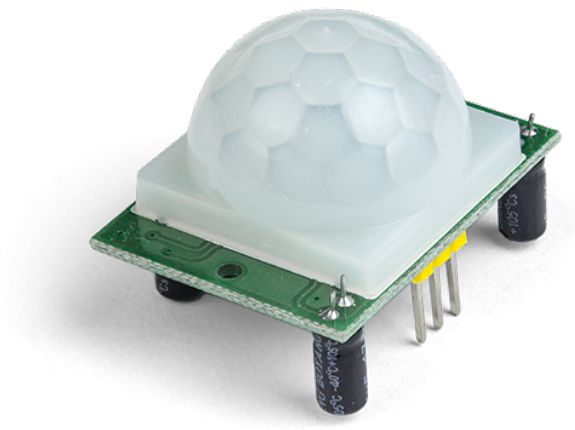
O sensor de presença e movimento PIR (*Passive Infra Red*) é um sensor infravermelho passivo, isto é, ele capta variações de irradiação de luz infravermelha, ou seja, variações de temperatura.

Todo objeto emite uma certa quantidade de luz infravermelha e quando ocorre uma mudança repentina no ambiente em que o sensor está instalado, como por exemplo a

passagem de uma pessoa na área de detecção, o sensor irá gerar um sinal elétrico e este sinal será captado pelo microcontrolador que realizará as ações programadas.

Neste projeto foi utilizado o módulo de controle automático HC-SR501, visualizado na Figura 14, que possui o sensor PIR integrado em uma placa de circuito impresso pronta para a conexão e funcionamento com um microcontrolador.

Figura 14 – Sensor de presença PIR HC-SR501



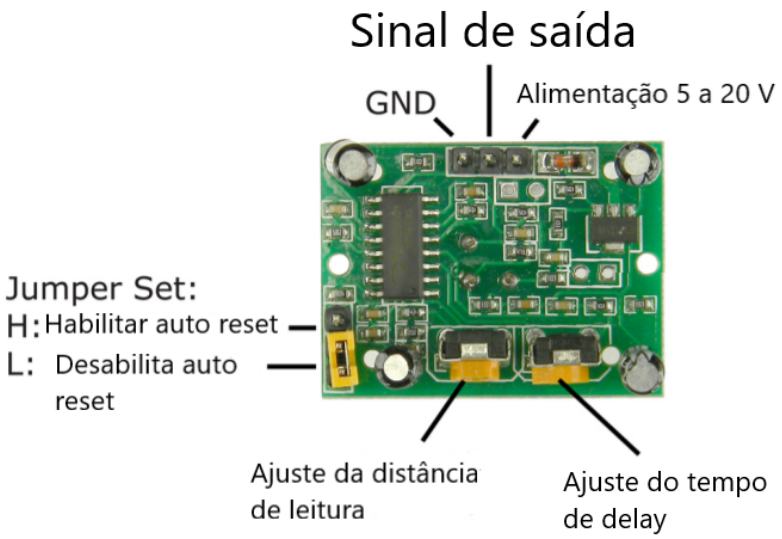
Fonte: Robocore (2021)

O módulo possui as seguintes características:

- Tensão de alimentação: 5 a 20 V;
- Corrente consumida: 65 mA;
- Tensão de saída 3,3 V;
- Tempo de delay: ajustável entre 5 a 300 s;
- Ângulo de abertura: até 120º;
- Distância máxima de leitura: ajustável entre 3 a 7 m

A pinagem do módulo HC-SR501 pode ser visualizada na Figura 15. O pino de alimentação é conectado a fonte CC (Corrente contínua) que alimenta o sistema, o pino GND é conectado a referência do sistema e o sinal de saída irá se conectar a porta lógica GPIO do NodeMCU.

Figura 15 – Relação de pinos HC-SR501



Fonte: Autoria própria

5.1.4 MÓDULO BUZZER

Em conjunto com o módulo HC-SR501, o módulo buzzer compõe a função de alarme do sistema implementado. Atuando como uma campainha vibratória passiva, sua porta de controle pode ser acionada tanto por um sinal digital quanto um sinal analógico com modulação PWM (*Pulse Width Modulation* - modulação por largura de pulso), sendo este último caso utilizado para controlar o volume e a frequência.

No sistema implementado, quando o sensor PIR envia um sinal para o NodeMCU (gerado pela detecção de presença ou movimento), este realiza um processamento e envia um comando ao Buzzer que funciona como atuador do sistema de alarme.

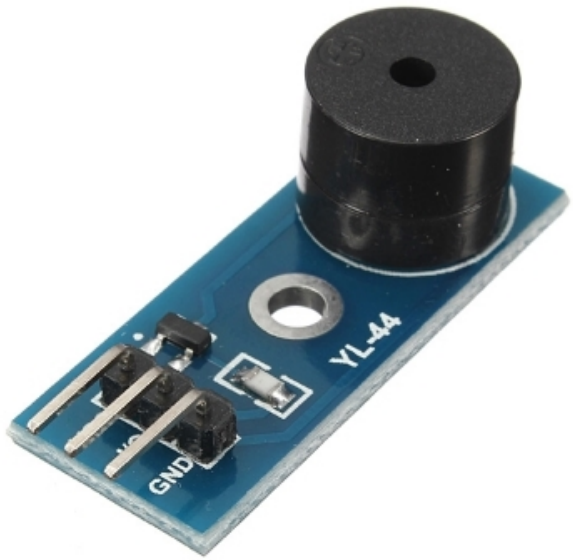
As especificações do módulo buzzer utilizado são as seguintes:

- Tensão de alimentação: 3,3 a 5 V;
- Alcance de frequência: próximo de 2 kHz;
- Corrente máxima consumida: 333 mA.

O módulo buzzer possui 3 pinos: o pino VCC que realiza a alimentação do módulo, o pino GND ligado à referência do sistema e o pino I/O (*Input/Output* - Entrada/Saída) que é conectado ao NodeMCU para ser realizado o controle do acionamento do módulo.

A Figura 16 apresenta o módulo buzzer utilizado no projeto e seus respectivos pinos: VCC, I/O e GND.

Figura 16 – Módulo buzzer



Fonte: Filipe Flop (2021)

5.1.5 MOTOR DE PASSO 28BYJ-48

Para o controle de abertura da persiana e do portão eletrônico foi utilizado o motor de passo 28BYJ-48. A escolha deste motor para a implementação no sistema se deve pelos seguintes motivos:

- Baixo custo de aquisição;
- Modelo mais popularizado da categoria;
- Fácil implementação;
- Possui um ótimo funcionamento em baixas velocidades;
- Entrega um torque elevado que é essencial para a aplicação;
- Possui funcionamento para os dois sentidos de giro.

Como pode ser visto na Figura 17, o motor de passo 28BYJ-48 possui uma saída com cinco fios. Os seguintes fios são ligados aos enrolamentos (fases): azul, rosa, amarelo e laranja. O fio vermelho conectado a porta comum (GND) ou referência do sistema.

Figura 17 – Motor de passo 28BYJ-48

Fonte: Filipe Flop (2021)

O motor de passo 28BYJ-48 possui as seguintes especificações:

- Tensão de alimentação: 5 V em corrente contínua;
- Número de fases: 4;
- Número de vias: 5;
- Caixa de redução 1/64;
- Ângulo de passo: 0,088°;
- Frequência: 100 Hz;
- Corrente consumida: até 500 mA.

Para o motor utilizado, têm-se os seguintes modos de operação:

- *Full Step*: passo completo com alto torque;
- *Wave Step*: passo completo com baixo torque;
- *Half Step*: meio passo;
- *Micro Step*: micro passo.

Como o motor foi utilizado para simular a abertura de persiana e um portão eletrônico, em que essas tarefas requerem um elevado torque com uma velocidade baixa e controlada, o modo de operação utilizado no projeto foi o *Full step*. Nesse modo de operação, duas fases são acionadas ao mesmo tempo em cada passo, porém a sequência dos passos deve ser seguida em uma ordem específica para que o motor mantenha o sentido de rotação e não perca velocidade e torque. A ordem a ser seguida está retratada na Figura 18, onde as linhas da tabela representam os passos e as colunas os enrolamentos (fases) do motor. O símbolo zero (0) representa a desenergização do enrolamento (0 V) e o símbolo um (1) representa a energização do enrolamento (+5 V).

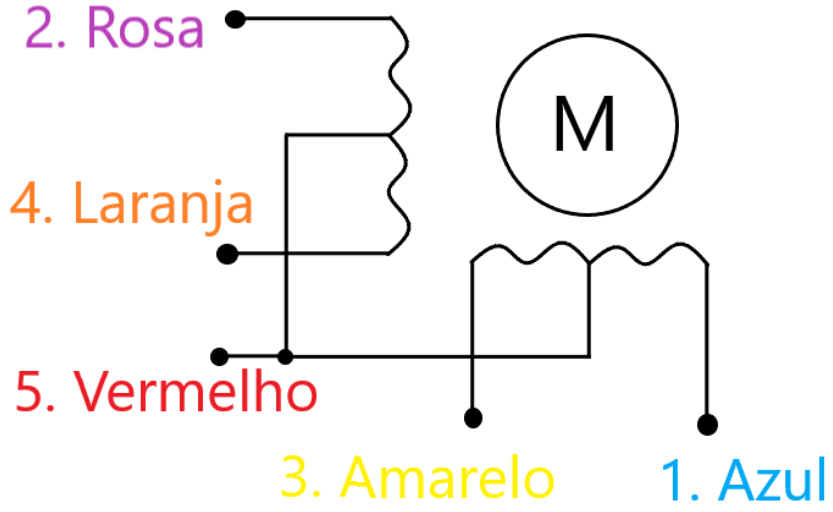
Figura 18 – Tabela *Full Step*

PASSO	ENROLAMENTOS			
	1	2	3	4
1	0	0	1	1
2	0	1	1	0
3	1	1	0	0
4	1	0	0	1

Fonte: Autoria própria

O diagrama das bobinas do motor pode ser visualizado na Figura 19.

Figura 19 – Diagrama de bobinas do motor de passo 28BYJ-48



Fonte: Autoria própria

5.1.6 MÓDULO DRIVER ULN2003

Como visto nas especificações do motor de passo 28BYJ-48, a corrente consumida pode chegar até 500 mA em seu funcionamento, porém a corrente máxima fornecida

pelas portas lógicas GPIO do NodeMCU é no máximo 12 mA. Dessa forma, para que seja possível realizar o acionamento e controle do motor, é necessário utilizar um módulo driver que amplifique a corrente de saída do NodeMCU. Para estas especificações de corrente, o módulo escolhido foi o modelo ULN2003.

Este módulo driver possui um conjunto de sete transistores do modelo *Darlington* que permitem o acionamento de cargas indutivas com a corrente fornecida.

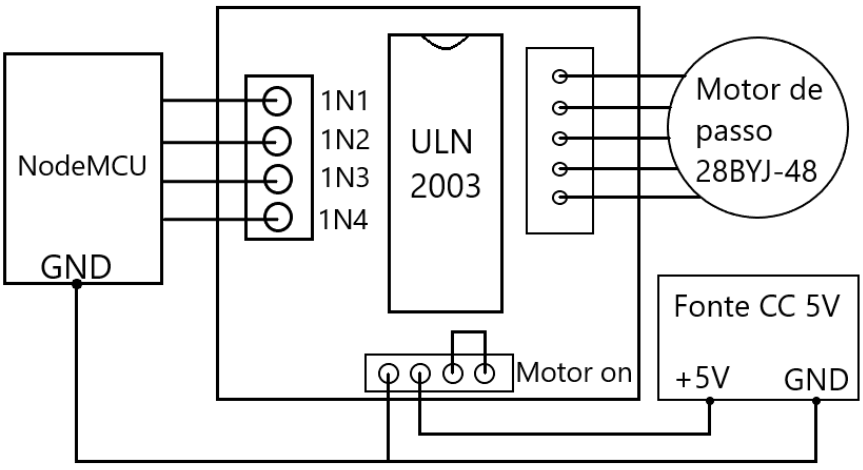
Além de possibilitar a alimentação e funcionamento do motor de passo, o módulo driver ULN2003 possui funcionamento de circuito em ponte H. Esse tipo de circuito é amplamente utilizado em projetos de automação, pois permite o controle do sentido da corrente que alimentará o motor, implicando no controle do sentido de rotação. O sentido de rotação de um motor CC se dá justamente pelo sentido da circulação da corrente que atravessa seu enrolamento, dessa forma, ao inverter o sentido da corrente, é invertido o sentido de rotação (ROMERO; BARBANO; MISOGUTI, 2019).

Esse módulo driver apresenta as seguintes especificações:

- Tensão de alimentação: 5 V;
- Corrente máxima de saída: 500 mA;
- Temperatura de operação: -40 a 85 °C
- Corrente máxima consumida: 1,2 mA

O diagrama do circuito contendo o motor de passo 28BYJ-48 e o NodeMCU conectados ao módulo driver ULN2003 está representado na Figura 20.

Figura 20 – Diagrama de montagem do circuito do motor de passo



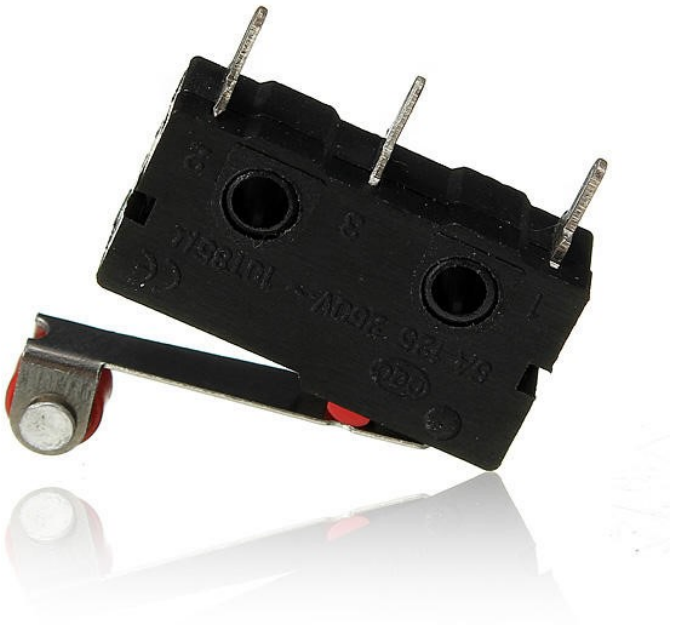
Fonte: Autoria própria

5.1.7 CHAVE FIM DE CURSO

Para auxiliar no controle do portão eletrônico e da persiana, foi utilizado o dispositivo visto na Figura 21 chamado chave fim de curso. Este é um componente que atua como uma chave de três vias. Possui uma haste que ao pressionar um botão no fim do seu curso, fecha um contato interno. O funcionamento das três vias se dá de maneira parecida ao módulo relé, onde uma das vias possui um contato normalmente aberto, outra via um contato normalmente fechado e a última via é a conexão comum entre os contatos. Quando ocorre o pressionamento do botão, a via comum fecha contato com a via normalmente aberta e abre contato com a via normalmente fechada. Quando o botão não é pressionado o caso inverso ocorre.

As chaves fim de curso foram utilizadas de maneira que atuassem como sensores para o portão eletrônico e a persiana. Elas foram dispostas nas extremidades de cada sistema e servem para monitorar se o portão e a persiana foram completamente abertos ou fechados. Dessa forma, caso as chaves forem pressionadas, elas enviam um sinal para o NodeMCU que monitora em *looping* este sinal enviado e atua sobre o controle dos motores de passo.

Figura 21 – Chave fim de curso



Fonte: Filipe Flop (2021)

5.1.8 GOOGLE FIREBASE

O *Firebase* é uma plataforma desenvolvida pela empresa *Google* que auxilia no desenvolvimento de aplicações *mobile* e *web*. Possui diversas funcionalidades que solucionam, de maneira rápida e simplificada, recursos necessários e comuns em uma aplicação, como

por exemplo: autenticação de usuário, armazenamento de dados, serviço de notificação, dentre outros.

A escolha de utilizar o *Firebase* como ferramenta auxiliar no projeto se deu por conta das seguintes vantagens:

- Fácil implementação e adequação com o sistema;
- Possui acesso gratuito através da versão *trial*;
- Suporte a diversos tipos de plataforma: iOS, Android, JavaScript e C++;
- Integração com diversas aplicações como as utilizadas neste trabalho: NodeMCU e Flutter;
- Possibilita o acesso aos dados armazenados de qualquer aparelho conectado a internet.

Neste trabalho, o *Firebase* tem como papel fundamental, o armazenamento dos dados e também a autenticação de usuário no aplicativo. Funcionando como uma ponte intermediária entre os blocos do sistema, o *Firebase* armazena os seguintes dados: comandos enviados pelo aplicativo, valores de temperatura e umidade coletados pelo sensor DHT11 e enviados pelo NodeMCU e cadastro dos usuários no aplicativo *mobile*.

5.1.9 FLUTTER

Flutter é uma estrutura de código aberto da empresa *Google* para construção de aplicativos multiplataformas, compilados nativamente a partir de uma única base de código.

É de extrema vantagem a utilização da plataforma *Flutter* para a criação de aplicações, pois os esforços podem ser concentrados inteiramente no código da aplicação, já que a estrutura permite compilar um único código para diferentes sistemas: iOS, Android, Windows, MacOS, Linux, Google Fuschia e Web.

5.2 METODOLOGIA

Uma pesquisa bibliográfica busca a resolução de um problema por meio de referenciais teóricos publicados (livros, jornais, artigos, etc.), analisando e discutindo as várias contribuições científicas. Esse tipo de pesquisa traz recursos para o conhecimento sobre o assunto pesquisado. Portanto, é de grande importância que o pesquisador realize um planejamento sistemático do processo de pesquisa, compreendendo desde a definição da problemática, passando pela sua construção lógica até a decisão de sua forma de comunicação e divulgação (BOCCATO, 2006).

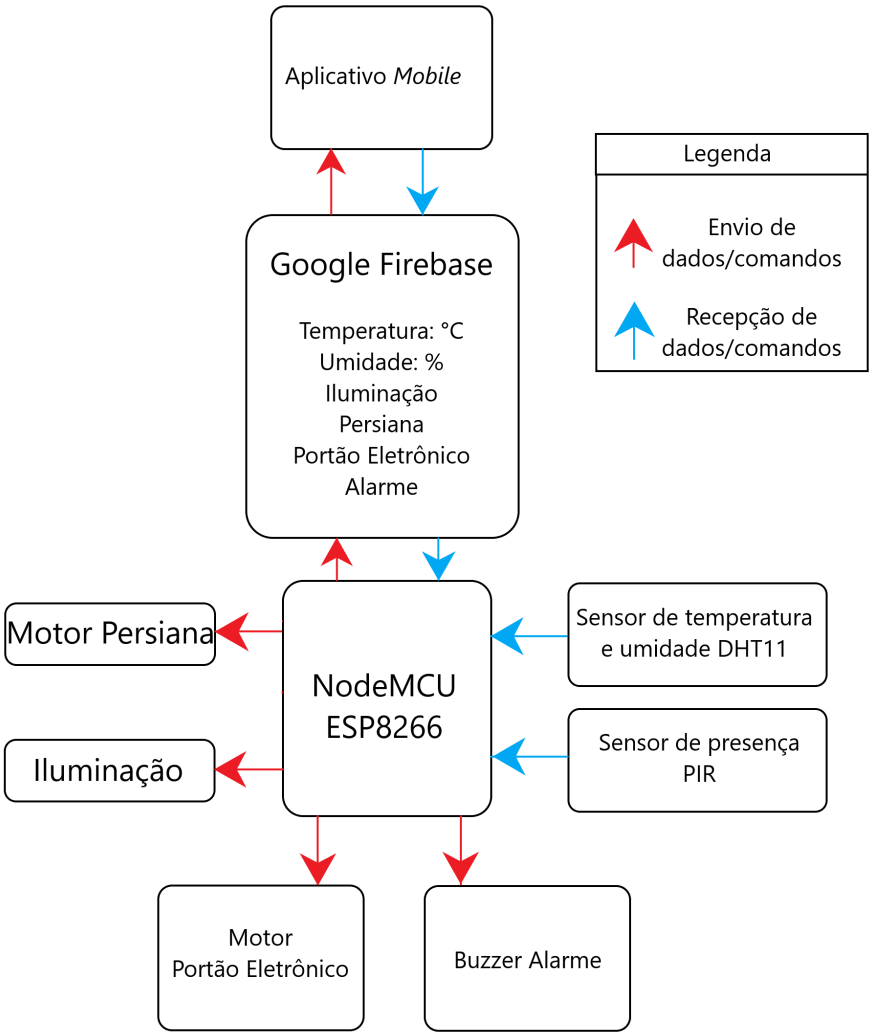
Para que fosse possível ser atingido o objetivo principal proposto pelo presente trabalho, foi adotado uma pesquisa de cunho exploratório utilizando-se de métodos experimentais e bibliográficos. Com isso em mente, foi primeiramente levantado um estudo

sobre o tema a partir de livros e artigos científicos para obter-se uma familiarização com os conteúdos abordados e em seguida foi traçado um planejamento sistemático sobre os assuntos a serem desenvolvidos. Este planejamento moldou os objetivos específicos do trabalho.

5.2.1 FUNCIONAMENTO DO SISTEMA

O sistema proposto tem como base para implementação o diagrama de blocos apresentado na Figura 22. Este diagrama será explicado nas próximas seções que são referentes a cada bloco que compõe o projeto.

Figura 22 – Diagrama de blocos do sistema implementado



Fonte: Autoria própria

5.2.1.1 CONSIDERAÇÕES INICIAIS

Para que seja possível o usuário final realizar o controle e monitoramento das variáveis do sistema é necessário uma UI (*user interface* - interface do usuário) que faça a conexão entre o restante do sistema e o próprio usuário. Neste projeto, a UI adotada foi um aplicativo *mobile* desenvolvido com a finalidade de atender aos requisitos do projeto de maneira intuitiva e otimizada.

Inicialmente, a UI desenvolvida foi uma *web page* que informava os dados de monitoramento e controle para o usuário. Porém, com o passar do desenvolvimento do projeto, foi notado que uma aplicação *mobile* se encaixava melhor ao tema proposto, já que a problemática gira em torno de uma automação residencial que gere conforto e acesso sem restrição de distância.

A *web page* era hospedada em um servidor criado pelo NodeMCU e isso restringia o acesso da página a somente usuários que estivessem conectados na mesma rede WiFi que o NodeMCU. Ao utilizar a UI instalada em um dispositivo móvel em conjunto com o *Firebase* é permitido ao usuário o acesso e controle das variáveis do sistema de qualquer lugar que esteja, desde que o dispositivo móvel possua uma conexão com a internet.

5.2.1.2 APLICATIVO MOBILE

Para o desenvolvimento do aplicativo, foi utilizado o editor de código fonte *Visual Studio Code* com a estrutura *Flutter* presente como *plugin* e a escrita do código foi dada na linguagem *Dart*.

A instalação do aplicativo nos dispositivos de sistema operacional Android é dada pelo arquivo do formato APK (*Android Application Pack*) compilado por um compilador externo chamado pela estrutura *Flutter* através do *Visual Studio Code*.

Ao abrir o aplicativo a tela inicial é a de *Login* onde usuários já cadastrados podem entrar e prosseguir para as telas seguintes. Já para usuários novos, está presente um botão que leva a uma tela de registo de novas contas onde nessa tela o usuário deve entrar com um endereço de email que ainda não foi cadastrado e que seja existente. A verificação da existência do email é feita pelo método *Auth* presente na biblioteca *FirebaseAuth* utilizada na programação do aplicativo.

Após realizar o *login*, o usuário é redirecionado para a tela principal de controle, onde é informado a temperatura em °C e umidade do local. Logo abaixo, são dispostos quatro botões.

O primeiro botão, denominado **Iluminação**, ao ser pressionado levará o usuário a tela de controle de iluminação da residência. Nessa tela estão presentes botões interativos para acionamento e desligamento das lâmpadas controladas. Os botões possuem dois estados para indicar ao usuário o estado de cada lâmpada.

O segundo botão, **Persianas**, leva o usuário a tela de controle das persianas. Nessa

tela, as persianas disponíveis para controle são identificadas com um botão de três estados, sendo cada estado indicado pelo ícone da sessão do botão. O usuário pode realizar o acionamento para abertura, fechamento ou então parar o motor das persianas no momento em que desejar.

O terceiro botão faz o controle do portão eletrônico, que ao ser pressionado uma vez ativa o motor do portão para realizar a abertura do mesmo. Caso pressionado novamente, antes do portão chegar ao fim do curso, o motor para. Ativando-o novamente faz com que o portão mude o sentido de atuação e passe a fechar. Se pressionar novamente antes que feche completamente, o motor para até o botão ser pressionado.

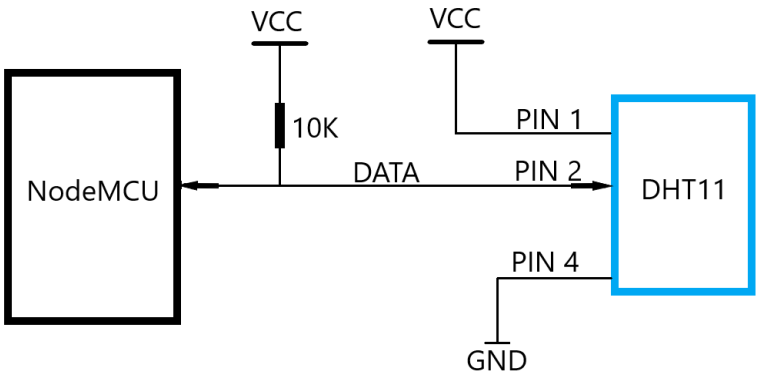
O último botão presente é um botão de dois estados referente ao alarme da casa. Caso esteja no estado Ligado, o monitoramento é realizado pelo sensor de presença PIR e, se identificado alguma movimentação de calor, o *Buzzer* é acionado. Contrariamente, se o estado do botão Alarme for Desligado, o sensor PIR não irá monitorar e portanto o *Buzzer* não irá ser acionado.

As telas presentes no aplicativo e que foram previamente citadas podem ser visualizadas no Apêndice A.

5.2.1.3 SISTEMA DE TEMPERATURA E UMIDADE

O esquema do circuito para funcionamento do sensor DHT11 é mostrado na Figura 23, onde se nota a utilização de um resistor de *pull-up* entre a conexão dos dados do sensor e a alimentação. É necessário a presença desse resistor para garantir que a entrada de dados recebe o nível lógico esperado. O *datasheet* do sensor DHT11 recomenda a utilização de um resistor entre 4,7 kΩ e 10 kΩ. Foi escolhido para este trabalho um resistor de 10 kΩ.

Figura 23 – Circuito esquematizado DHT11



Fonte: Autoria própria

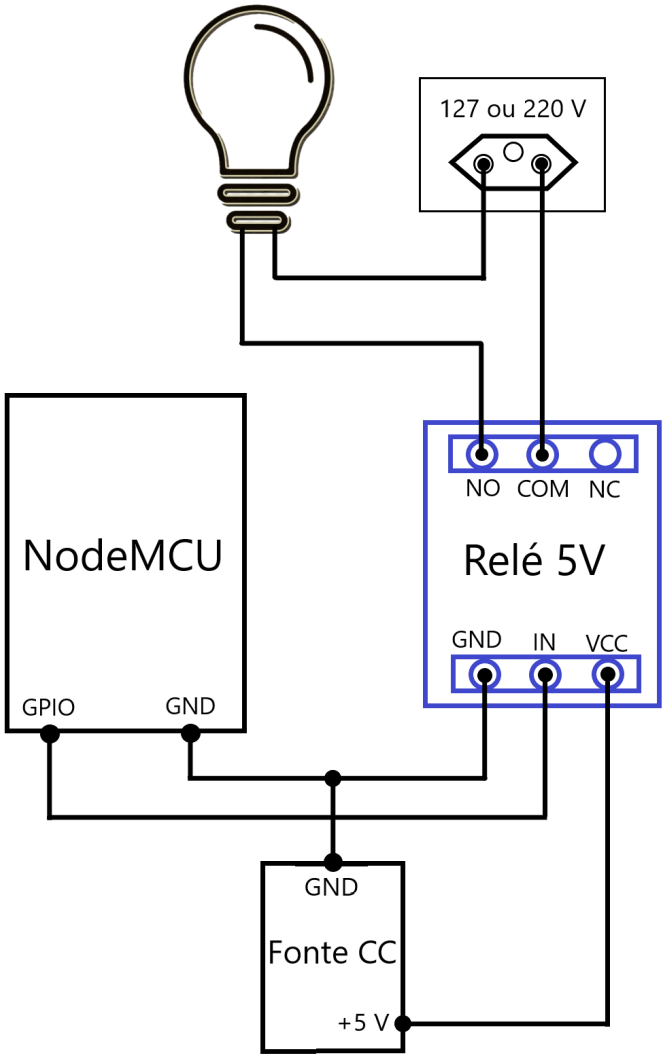
O NodeMCU recebe constantemente os dados referentes à temperatura e umidade lidos pelo sensor DHT11 e este se comunica com o servidor *Firebase* para a atualização

dos dados presentes no banco de armazenamento. O aplicativo por sua vez, comunica-se com o *Firebase* buscando realizar uma leitura dos dados de temperatura e umidade, para que a UI apresente os dados lidos sempre atualizados.

5.2.1.4 SISTEMA DE ILUMINAÇÃO

O diagrama do circuito de controle da iluminação está esquematizado na Figura 24. Os módulos relés são alimentados por uma fonte CC de 5V e o pino de GND é conectado à referência do sistema. O pino de controle dos módulos relés são conectados às portas lógicas GPIO do NodeMCU, sendo que estas atuarão como saídas. Um terminal da lâmpada é conectado diretamente a fase da alimentação de 127 V (nesse caso foi utilizada alimentação monofásica da rede) e o outro terminal é conectado ao pino de saída do relé NO (*Normally Open* - normalmente aberto). O neutro da alimentação da rede é conectado a porta comum (COM) do relé.

Figura 24 – Circuito esquematizado do controle de Iluminação



Fonte: Autoria própria

O acionamento da iluminação parte do aplicativo, onde através da UI o usuário aciona o comando para ligar ou desligar uma determinada lâmpada e esse comando atualiza um dado no *Firebase* referente a lâmpada escolhida. Este dado em questão é lido constantemente pelo NodeMCU que por sua vez envia um sinal lógico para a bobina do relé. Caso o dado lido indique a energização da lâmpada, o sinal enviado pelo NodeMCU irá comutar os contatos do relé fazendo com que a lâmpada seja ligada.

Para que seja possível acionar diversas lâmpadas separadamente, cada uma deve ser controlada por um relé próprio. Caso o acionamento for comum a duas lâmpadas ou mais, a utilização de apenas um relé é suficiente para o devido funcionamento da automatização da iluminação, pois este único relé será responsável pelo controle da energização das lâmpadas.

5.2.1.5 SISTEMA DE CONTROLE DAS PERSIANAS

O diagrama do circuito utilizado para o controle das persianas está esquematizado na Figura 20. O motor de passo possui os condutores que realizam a alimentação das bobinas e o condutor de referência conectados aos pinos de saída do driver ULN2003. O driver por sua vez é alimentado por uma fonte CC 5 V externa e seu pino GND é conectado a referência do sistema. Os pinos de *Motor ON* devem permanecer curto-circuitados para que o motor possa ser ligado quando as bobinas forem energizadas. As entradas (1N1, 1N2, 1N3 e 1N4) são conectadas as portas lógicas GPIO do NodeMCU. Essas entradas recebem sinais lógicos que comandam a ordem do acionamento das bobinas do motor de passo.

O controle das persianas começa com o comando do usuário através do aplicativo. Ao selecionar um dos três estados possíveis para as persianas, o aplicativo atualiza um dado (correspondente a persiana em questão) armazenado no *Firebase*. Este dado, que é monitorado constantemente pelo NodeMCU através da função *loop* de sua programação, quando verificado uma alteração no seu estado, o NodeMCU envia para o driver ULN2003 uma sequência de sinais lógicos que corresponderão aos passos que o motor de passo deve executar de acordo com o comando escolhido pelo usuário. Assim, ao selecionar um dos três estados para as persianas no aplicativo, o usuário poderá estar ligando o motor em um sentido de rotação, invertendo o sentido de rotação ou então desligando o motor.

5.2.1.6 SISTEMA DE CONTROLE DO PORTÃO ELETRÔNICO

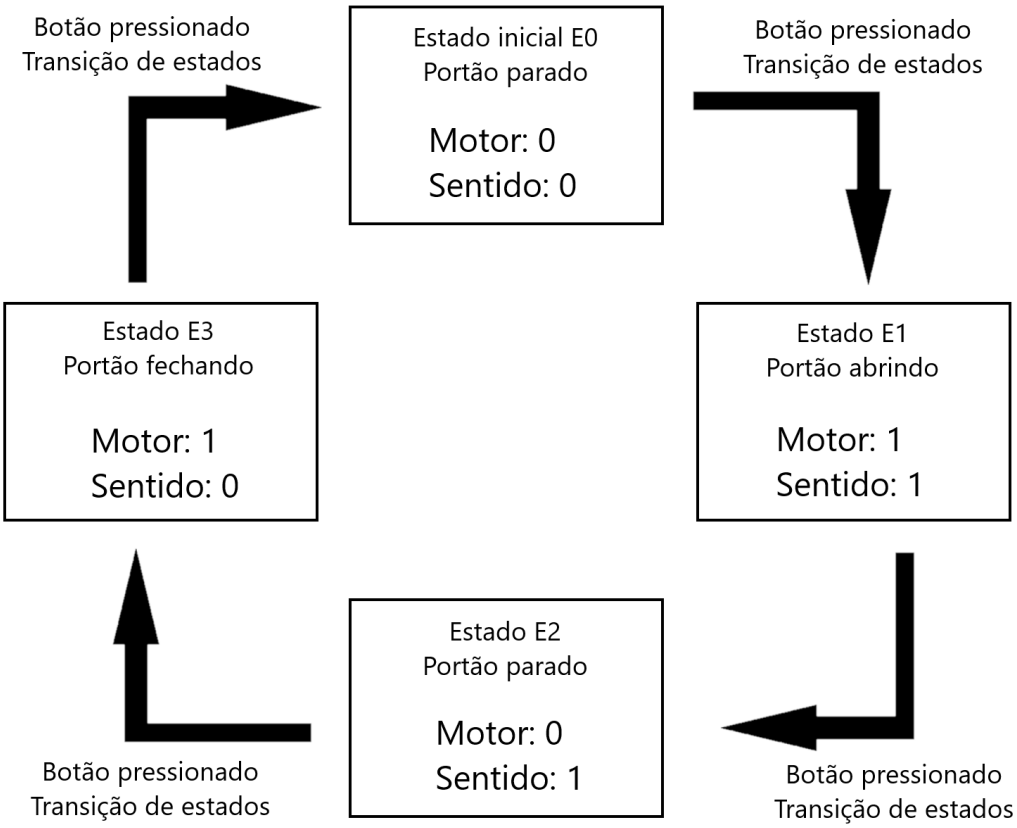
Semelhante ao controle das persianas, o diagrama do circuito utilizado para o controle do portão eletrônico é representado na Figura 20, ou seja, o circuito será disposto de maneira muito parecida, diferindo apenas nas conexões das entradas do driver, sendo que estas, no sistema de controle do portão eletrônico, serão conectadas a outras quatro portas GPIO diferentes das utilizadas no sistema de controle das persianas.

A grande diferença do sistema do portão eletrônico para o das persianas está em como o controle é realizado. O controle do sistema do portão eletrônico é feito utilizando o conceito de máquinas de estado.

Máquina de estados é um modelo de comportamento composto de um número finito de estados, onde estão inclusos as ações e as transições entre estados. A máquina pode estar somente em um estado por vez, sendo este estado chamado de estado atual. Cada estado representa uma situação relevante do sistema. Os estados carregam informações do estado anterior para que, quando ocorrerem as transições, as ações do estado atual sejam baseadas nas informações do estado anterior. Transições são as mudanças entre estados e as ações são responsáveis por realizar um determinado processamento.

O diagrama representando a máquina de estados implementada ao sistema de controle do portão eletrônico está ilustrado na Figura 25.

Figura 25 – Diagrama da máquina de estados implementada ao portão eletrônico



Fonte: Autoria própria

Para explicar o funcionamento da máquina de estados empregada, dois dados foram controlados, sendo estes: **Motor** que indica a energização do motor de passo (0: desligado ; 1: ligado) e **Sentido** que indica o sentido de rotação do motor de passo (0: sentido horário; 1: sentido anti-horário).

Quando o sistema é ligado, por padrão, o portão deverá estar fechado e parado, assim

é definido o estado inicial E0 da presente máquina de estados. Neste estado é verificado que tanto **Motor** quanto **Sentido** estão com valores atribuídos em zero. Somente este estado pode possuir essas indicações dos dados, pois em uma máquina de estados finitos, cada estado há de ser único e exclusivo. A máquina de estados permanecerá no estado atual até que seja acionado uma transição, no caso quando o botão **Portão Eletrônico** é pressionado na UI do aplicativo. Ao ser pressionado o botão, ocorre a transição do estado E0 para E1. O portão passa a abrir e os dados indicadores são **Motor**: 1 e **Sentido**: 1.

É verificado dessa forma uma sequência específica para a máquina de estados desenvolvida, sendo essa ordem sequencial: E0, E1, E2, E3, E0, E1, E2, E3, ... (repetindo-se infinitamente enquanto o sistema estiver operando). Ao chegar no estado E3, quando ocorrer a próxima transição, a máquina de estados volta ao estado inicial E0 e a sequência é retomada.

Os dados **Motor** e **Sentido** estão armazenados no *Firebase* e seus valores são atualizados conforme o botão **Portão Eletrônico** é pressionado pelo usuário no aplicativo. O NodeMCU monitora constantemente esses dados e quando notado uma transição de estado é enviado um respectivo sinal, pelas portas GPIO, ao driver que acionará o motor de passo.

Em um sistema, quando há a utilização de botões (mecânicos ou implementados em *softwares*) surge um fenômeno inerente a estes objetos chamado *bouncing*, o qual consiste na geração de múltiplos sinais de saída ao realizar um contato prolongado com o botão. Esses sinais indesejados podem gerar erro na leitura dos dados causando um mal funcionamento do sistema. Dessa forma, para solucionar este problema, foi implementado um algoritmo de *Debouncing* (técnica que trata e soluciona o problema de *bouncing*) na programação do botão **Portão Eletrônico**, presente no aplicativo, *mobile* que detecta apenas a primeira ativação do botão (borda de subida), fazendo com que não sejam gerados os múltiplos sinais indesejados.

5.2.1.7 SISTEMA DE CONTROLE DO ALARME

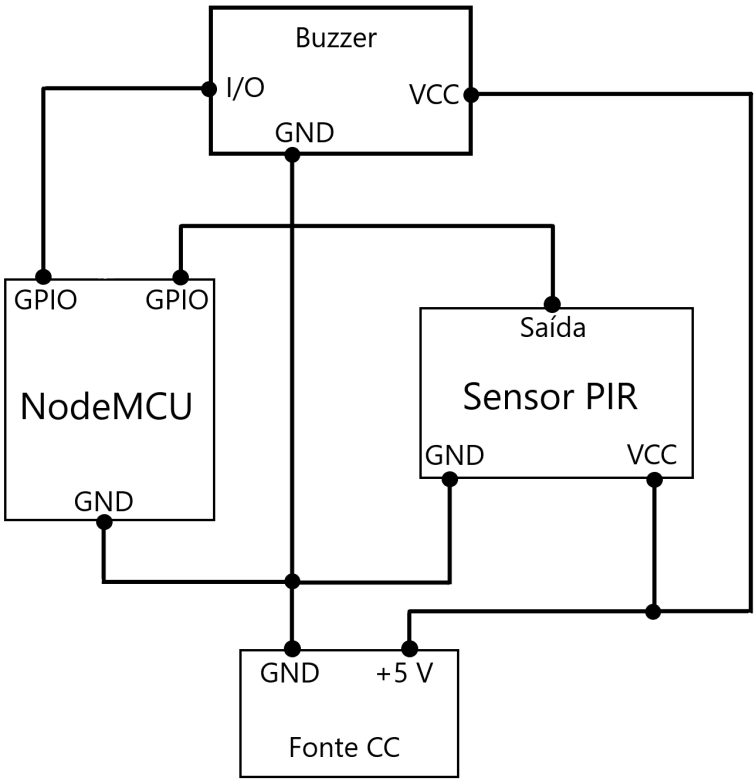
O diagrama do circuito do sistema de alarme é ilustrado na Figura 26.

Ambos os dispositivos, sensor PIR e Buzzer possuem a alimentação VCC provinda de uma fonte CC 5V externa, o pino GND conectado à referência do sistema e o pino de controle conectado a uma porta lógica GPIO do NodeMCU.

O controle do alarme tem início com a ativação do botão de dois estados presente no aplicativo: **Alarme** (**Ligado** ; **Desligado**). Um dado referente ao estado do botão **Alarme** é armazenado no *Firebase* e esse dado é atualizado conforme o estado do botão é alterado. O NodeMCU monitora constantemente esse dado juntamente com o sinal obtido do sensor PIR. Caso o botão esteja no estado **Ligado** e o sensor PIR enviar um sinal de detecção de movimento/presença, o NodeMCU ativará o módulo buzzer gerando um sinal sonoro para efeito de alarme. Caso o botão esteja no estado **Ligado** e não acontecer

detecção pelo sensor PIR, o buzzer não irá ser ativado. Da mesma maneira, caso o botão esteja no estado **Desligado** e acontecer uma detecção pelo sensor PIR, o buzzer não irá ser ativado.

Figura 26 – Diagrama do circuito do sistema de alarme



Fonte: Autoria própria

5.2.2 CIRCUITO

Para a montagem do circuito foram utilizados os seguintes componentes:

- NodeMCU ESP8266;
- Matriz de contato (*protoboard*);
- Cabos *jumpers*;
- Sensor de temperatura e umidade DHT11;
- Módulo com sensor de presença PIR HC-SR501;
- Um motor de passo 28BYJ-48;
- Módulo buzzer;
- Módulo driver ULN2003;

- Duas chaves fim de curso;
- Resistor 10 kΩ;
- Fonte CC de 5 V;
- Cabo micro USB para programação do NodeMCU

5.2.3 REPRESENTAÇÃO RESIDENCIAL

Analisando a dificuldade que seria implementar um sistema de automação proposto em um trabalho de conclusão de curso a uma residência real, foi concluído que a automação de uma maquete, que representa uma residência, iria simular de maneira satisfatória a execução do projeto. Dessa forma, foi projetada uma maquete residencial simples que possui uma pequena divisão de cômodos e que abriga os componentes necessários para a implementação prática do sistema.

A planta projetada e as imagens da maquete finalizada estão contidas no Apêndice C. A maquete foi construída utilizando recortes de material MDF (*Medium Density Fiberboard*). Estes recortes foram fixados a uma base e uns aos outros com parafusos, assim formando a representação da divisão dos cômodos.

5.2.4 PROGRAMAÇÃO: NODEMCU

Os códigos referentes a programação do NodeMCU estão contidos no Apêndice B.

Para começar a programação do chip ESP8266 deve-se incluir as respectivas bibliotecas na IDE do Arduino.

Algumas constantes foram definidas logo no escopo do código, tais como: modelo do sensor DHT, nome e senha da rede Wifi que será conectada, nome do projeto criado no servidor *Firebase* e chave de autenticação para acesso do banco de dados do *Firebase*.

Em seguida, foram criadas variáveis que irão receber os valores de temperatura, umidade, pinos de iluminação, pinos dos motores, pino do sensor de presença PIR, pino do buzzer e do sensor DHT11.

Com o escopo definido, cria-se um objeto do tipo DHT que receberá o valor do pino utilizado na transferência de dados e o modelo do sensor.

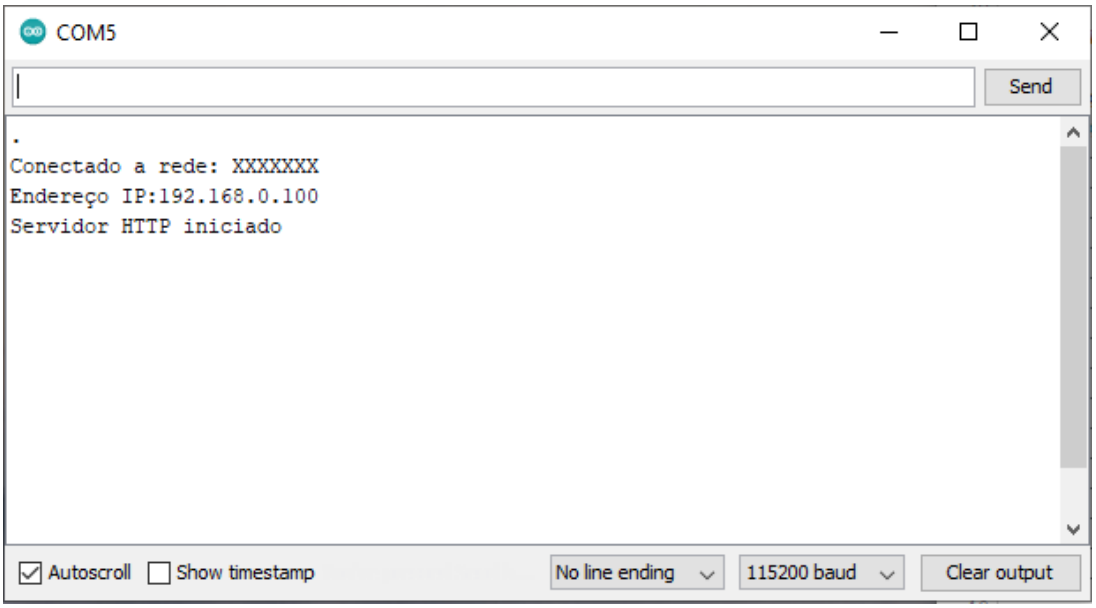
Dentro da rotina principal **void setup()** são realizados os seguintes passos:

- É inicializado, através da função **serial.begin()**, o monitor serial, contido na Figura 27, para o monitoramento do sistema;
- Definida as entradas e saídas dos pinos GPIO do NodeMCU;
- Inicializada a conexão entre o NodeMCU e a rede Wifi;
- Inicializada a conexão com a base de dados do servidor *Firebase*.

A rotina de repetição **void loop()**, possui as seguintes instruções:

- Leitura constante da temperatura através da função **dht.readTemperature()**;
- Leitura constante da umidade pela função **dht.readHumidity()**;
- Leitura constante da detecção do sensor de presença PIR;
- Leitura constante dos dados resgatados do servidor *Firebase*;
- Acionamento dos circuitos terminais conforme a leitura dos dados resgatados;
- Chamada da função de controle de passos dos motores de passo.

Figura 27 – Monitor serial



Fonte: Autoria própria

6 RESULTADOS

Para facilitar o entendimento do leitor, os resultados serão distribuídos em seções, cada uma comentando um tópico específico do trabalho.

6.1 APLICATIVO

O desenvolvimento do aplicativo utilizando a plataforma *Flutter* gerou resultados satisfatórios no que se diz respeito à funcionalidade das interfaces, resposta dos comandos e funcionalidades que tornam o aplicativo intuitivo e com uma estética agradável ao usuário. O fato de ser uma plataforma que está se tornando popular constantemente e que possui uma densa quantidade de material para ser integrado, fez com que o tempo para aprendizado da ferramenta e da linguagem *Dart* fosse reduzido.

O aplicativo possui um sistema funcional de cadastro e *login* de usuários, porém não aprimorados.

A sua resposta aos comandos do usuário são rápidas, suaves e interativas. A comunicação com a base de dados hospedada no servidor da plataforma *Firebase* é rápida, não apresentou perda de dados, funcionou como o esperado em ambos sentidos de tramitação de dados, isto é, tanto no envio como na aquisição dos valores armazenados.

Uma situação que não funciona da maneira esperada acontece quando ocorre a mudança das telas de controle da iluminação ou de controle das persianas para a tela principal, em que os indicadores visuais dos valores não são mantidos, porém os valores dos dados no *Firebase* não são perdidos.

O objetivo não alcançado com o aplicativo foi de torná-lo multiplataforma, isto é, disponível tanto para dispositivos com sistema operacional Android quanto para iOS.

6.2 CIRCUITO

Todos os circuitos utilizados para o controle de cada sistema isolado funcionaram de maneira correta, sem apresentar variações bruscas nas medições, funcionando dentro dos valores esperados e com rápida resposta aos comandos enviados pelo NodeMCU.

Ao serem conectados ao NodeMCU, os circuitos mantiveram suas respostas inalteradas no que se diz respeito a funcionalidade final, porém algumas variações nas medições dos sensores e um *delay* no acionamento das lâmpadas surgiu.

Devido a limitada quantidade de portas lógicas GPIO presentes no NodeMCU, o sistema completo final possui poucos componentes disponíveis em cada segmento (três lâmpadas, um alarme, um módulo buzzer, um motor de passo e um sensor de temperatura e umidade). Pensando nisso, para que o sistema proposto se torne viável de ser implementado a uma residência, vários componentes NodeMCU podem ser conectados em

paralelo para atender ao mesmo sistema, cada um realizando o controle e monitoramento de um específico segmento. Além da viabilidade, possuir vários componentes NodeMCU irá otimizar o sistema como um todo, pois, o tempo de resposta gerado por cada função de monitoramento inerente ao microcontrolador será reduzido, evitando que ocorra eventuais percas de dados ou comandos.

Apesar disso, o alcance do objetivo proposto pelo trabalho não foi prejudicado, pois, a adição de componentes em cada sistema desenvolvido implicaria basicamente no mesmo conceito e funcionamento, porém de forma otimizada em comparação ao sistema resultante.

6.3 TRABALHOS FUTUROS

Para eventuais trabalhos futuros que abordem o tema do presente trabalho, segue algumas sugestões de melhorias que podem ser realizadas para complementação e aperfeiçoamento do projeto:

- Desenvolver uma placa de circuito impresso (PCB) que possua todos os sistemas integrados;
- Utilizando da PCB desenvolvida, projetar um compartimento onde fique instalado o sistema completo, possuindo bornes e terminais que facilitem as conexões com o sistema elétrico da residência;
- Projetar uma fonte CC de 5 V que realize a alimentação de todo o sistema a partir da alimentação da rede;
- Modificar o aplicativo *mobile* para que a UI seja específica a cada usuário cadastrado;
- Expandir o uso do aplicativo para dispositivos com sistema operacional iOS;
- Utilizar dois componentes NodeMCU para habilitar um maior número de portas lógicas GPIO e, consequentemente, aumentar o número de dispositivos a serem controlados;
- Implementar o projeto utilizando outra plataforma de controle (microcontrolador), como por exemplo: ESP32;
- Implementar o sistema a uma residêncial real.

7 CONCLUSÃO

O monitoramento dos dados de temperatura e umidade e o controle de dispositivos presentes no ambiente são funcionalidades imprescindíveis quando o assunto é automação residencial visando conforto e praticidade. Aliar a possibilidade de controle na palma das mãos (*smartphones*) a um baixo custo de implementação, torna o tema ainda mais abrangente e interessante de ser estudado. Pensando nisso, o objetivo do trabalho de desenvolver um sistema completo que automatize uma residência através do conceito de IoT foi atingido com sucesso.

Com o desenvolvimento deste trabalho foi possível visualizar a acessibilidade e praticidade alta da implementação de um sistema de automação, sobretudo com a utilização da plataforma NodeMCU, que se mostrou uma excelente alternativa às convencionais plataformas de IoT presentes no mercado.

Utilizando-se componentes simples e de baixo custo, facilmente encontrados no mercado, junto com ferramentas gratuitas para desenvolvimento de aplicações e armazenamento de dados, foi possível realizar em uma maquete de representação residencial um sistema de automação que pode ser expandido e adaptado para uma residência real.

Os temas que circundam os conceitos de IoT e automação residencial se expandem diariamente e, consequentemente, se popularizam em ambientes cada vez menos tecnológicos, ou seja, tornam-se acessíveis a uma maior quantidade de pessoas. Assim, a criação e publicação de conteúdos científicos, como o presente trabalho, contribuem para a disseminação e consequente evolução do assunto abordado, sendo um aspecto vantajoso tanto para a comunidade acadêmica/científica, quanto para a sociedade.

REFERÊNCIAS

ALBERTIN, A. L.; ALBERTIN, R. M. de M. A internet das coisas irá muito além as coisas. **GV EXECUTIVO**, v. 16, n. 2, p. 12–17, 2017.

BOCCATO, V. R. C. Metodologia da pesquisa bibliográfica na área odontológica e o artigo científico como forma de comunicação. **Rev. Odontol. Univ. Cidade São Paulo, São Paulo**, v. 18, n. 3, p. 265–274, 2006.

BRAGA, N. C. **Relés: Circuitos e aplicações**. [S.l.]: Editora Newton C. Braga, 2017.

BRITO, F. **Sensores e atuadores**. [S.l.]: Saraiva Educação SA, 2014.

GOTARDO, R. A. Linguagem de programação. **Ed. Rio de**, p. 17, 2015.

JUNIOR, J. H. B.; LIMA, M. L.; OLIVEIRA, T. M. de. Internet das coisas, um conceito em construção. **Revista Científica Redes de Computadores**, v. 1, n. 1, p. 12–16, 2018.

MONTENEGRO, L. P. Controle digital de velocidade de um motor cc de baixa potência utilizando arduino. 2021.

PATSKO, L. F. Tutorial: Aplicações, funcionamento e utilização de sensores. **Maxwell Bohr: Instrumentação eletrônica**, p. 41, 2006.

PATSKO, L. F. Tutorial controle de motor de passo. **Maxwell Bohr–Instrumentação Eletrônica**, 2006.

PIRES, P. F. et al. Plataformas para a internet das coisas. **Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, p. 110–169, 2015.

ROMERO, A. L.; BARBANO, E. C.; MISOGUTI, L. Sistema computadorizado para deslocamento de amostra com motor de passo utilizando o l298: aplicação na técnica de varredura-z. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 41, 2019.

SAIKRISHNA, M.; VIJAYKIRAN, G. Iot based home electrical appliances control using node mcu. **International Journal of Scientific Engineering and Technology Research, ISSN**, p. 2319–8885, 2017.

SANTOS, B. P. et al. Internet das coisas: da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, v. 31, 2016.

SANTOS, V. P. Vinicius Puga de A. Motor de passo. 2008.

SCHILDT, H. **C completo e total**. [S.l.]: Makron, 1997.

SEIXAS, M.; CONTINI, E. Internet das coisas (iot): inovação para o agronegócio. **Área de Informação da Sede-Nota Técnica/Nota Científica (ALICE)**, Brasília, DF: Secretaria de Inteligência e Macroestratégia, 2017., 2017.

SIPANI, J. P. et al. Wireless sensor network for monitoring & control of environmental factors using arduino. **International Journal of Interactive Mobile Technologies (iJIM)**, v. 12, n. 2, p. 15–26, 2018.

THAMARAIMANALAN, T. et al. Smart garden monitoring system using iot. In: . [S.l.: s.n.], 2018.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. de. **Sensores industriais: fundamentos e aplicações**. [S.l.]: Saraiva Educação SA, 2020.

TRINDADE, R. S.; ALMEIDA, S. G. M.; PINTO, P. R. Implementacao de uma rede neural com microcontrolador pic. 2009.

VANAJA, K. J. et al. Iot based agriculture system using node mcu. **International Research Journal of Engineering and Technology**, v. 5, n. 3, p. 3025–3028, 2018.

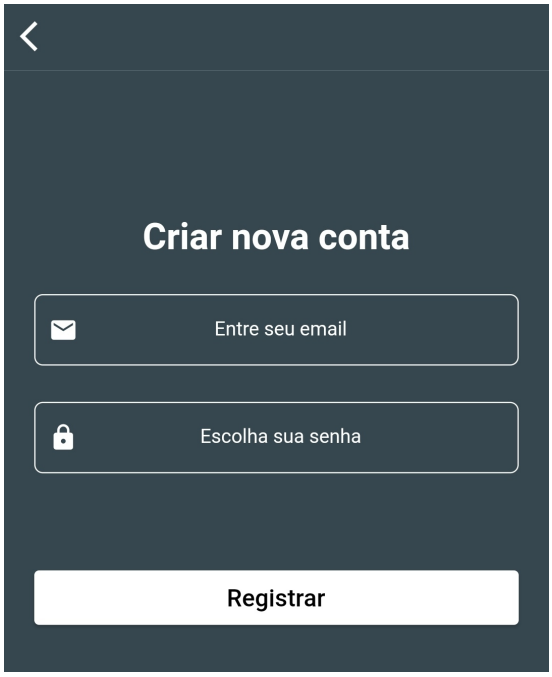
WENDLING, M. Sensores. **Universidade Estadual Paulista. São Paulo**, v. 2010, p. 20, 2010.

APÊNDICE A – TELAS DO APLICATIVO

Figura 28 – Tela inicial do aplicativo

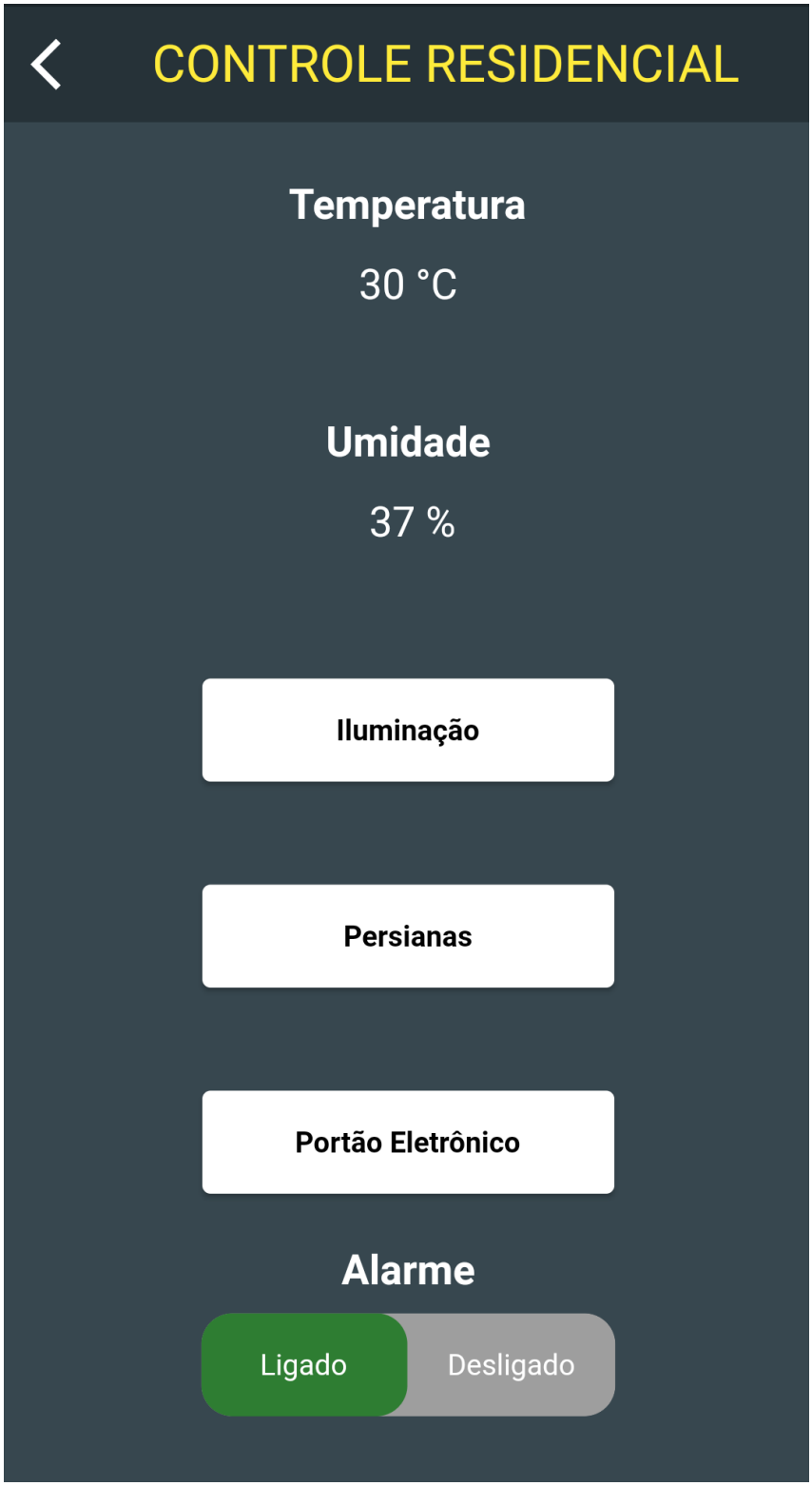
Fonte: Autoria própria

Figura 29 – Tela de registro de novas contas



Fonte: Autoria própria

Figura 30 – Tela principal do aplicativo



Fonte: Autoria própria

Figura 31 – Tela para controle da iluminação

Fonte: Autoria própria

Figura 32 – Tela para controle de persianas

Fonte: Autoria própria

APÊNDICE B – CÓDIGOS NODEMCU

Quadro 1 – Bibliotecas utilizadas

```
1 #include <ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <ESP8266WebServer.h>
4 #include <DHT.h>
5 #include <FirebaseArduino.h>
6 #include <ArduinoJson.h>
```

Fonte: Autoria própria

Quadro 2 – Constantes definidas

```
1  #define ssid      "XXXXXXX" //Nome da rede Wifi
2  #define password  "XXXXXX" //Senha da rede
3  #define DHTTYPE   DHT11 //Define o modelo de sensor utilizado
4  #define FIREBASE_HOST  "XXXX" //Especifica o projeto Firebase
5  #define FIREBASE_AUTH  "XXXX" //Chave de autenticação Firebase
6  #define IN1 D0 //Conexão do motor de passo
7  #define IN2 D1 //Conexão do motor de passo
8  #define IN3 D2 //Conexão do motor de passo
9  #define IN4 D3 //Conexão do motor de passo
10 #define PIR D4 //Pino de dados sensor PIR
11 #define BUZZER D5 //Pino de controle buzzer
12 #define DHT D6 //Pino de dados sensor DHT
13 #define LAMPADA1 D7 //Pino de controle lâmpada 1
14 #define LAMPADA2 D8 //Pino de controle lâmpada 2
```

Fonte: Autoria própria

Quadro 3 – Criação dos objetos

```
1 DHT dht(DHTPIN, DHTTYPE); //criação do objeto tipo DHT
```

Fonte: Autoria própria

Quadro 4 – Criação de variáveis

```
1 float   t = 0 ; //valor de temperatura
2 float   u = 0 ; //valor de umidade
3 int chave1 = 3 //GPIO3 para conexão da chave fim de curso 1
4 int chave1 = 1 //GPIO1 para conexão da chave fim de curso 2
```

Fonte: Autoria própria

Figura 33 – Rotina para controle de passo (*Full step*) no sentido horário do motor de passos

```
void passoCompletoHorario(){
  {
    switch(passo){
      case 0:
      digitalWrite(Pin1, LOW);
      digitalWrite(Pin2, LOW);
      digitalWrite(Pin3, HIGH);
      digitalWrite(Pin4, HIGH);
      break;

      case 1:
      digitalWrite(Pin1, LOW);
      digitalWrite(Pin2, HIGH);
      digitalWrite(Pin3, HIGH);
      digitalWrite(Pin4, LOW);
      break;

      case 2:
      digitalWrite(Pin1, HIGH);
      digitalWrite(Pin2, HIGH);
      digitalWrite(Pin3, LOW);
      digitalWrite(Pin4, LOW);
      break;

      case 3:
      digitalWrite(Pin1, HIGH);
      digitalWrite(Pin2, LOW);
      digitalWrite(Pin3, LOW);
      digitalWrite(Pin4, HIGH);
      break;

    }

     passo++;
     if(passo > 3){
     passo = 0;
     }
}
```

Fonte: Autoria própria

Figura 34 – Rotina para controle de passo (*Full step*) no sentido anti-horário do motor de passos

```
void passoCompletoAntiHorario(){
      switch(passo){
      case 0:
      digitalWrite(Pin1, HIGH);
      digitalWrite(Pin2, LOW);
      digitalWrite(Pin3, LOW);
      digitalWrite(Pin4, HIGH);
      break;

      case 1:
      digitalWrite(Pin1, HIGH);
      digitalWrite(Pin2, HIGH);
      digitalWrite(Pin3, LOW);
      digitalWrite(Pin4, LOW);
      break;

      case 2:
      digitalWrite(Pin1, LOW);
      digitalWrite(Pin2, HIGH);
      digitalWrite(Pin3, HIGH);
      digitalWrite(Pin4, LOW);
      break;

      case 3:
      digitalWrite(Pin1, LOW);
      digitalWrite(Pin2, LOW);
      digitalWrite(Pin3, HIGH);
      digitalWrite(Pin4, HIGH);
      break;
    }

     passo++;
     if(passo > 3){
     passo = 0;
     }
}
```

Fonte: Autoria própria

APÊNDICE C – MAQUETE PARA REPRESENTAÇÃO RESIDENCIAL

Figura 35 – Planta da maquete residencial

Fonte: Autoria própria

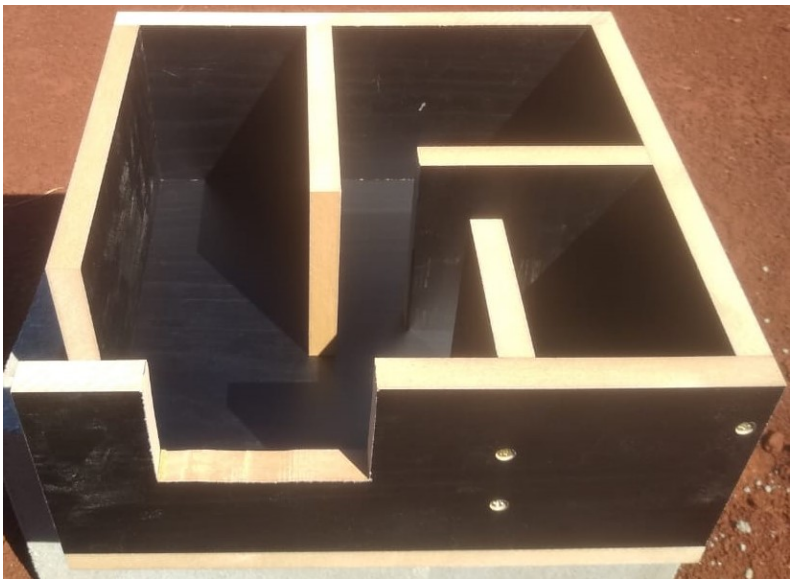
Figura 36 – Vista 1 da maquete

Fonte: Autoria própria

Figura 37 – Vista superior da maquete

Fonte: Autoria própria

Figura 38 – Vista 2 da maquete



Fonte: Autoria própria